**Proje Ana Alanı : YAZILIM**

**Proje Tematik Alanı : STEAM**

# Proje Adı : CANLI POPÜLASYON DİNAMİKLERİNİN VE DEĞİŞİMLERİNİN BİLGİSAYAR ORTAMINDA SİMÜLASYONU


## Özet

Bu projenin amacı, farklı durum ve koşullarda araştırma konusu canlıların vereceği tepkileri ve popülasyonlarının hangi şartlarda nasıl değişimler göstereceğini gözleme imkânı yaratacak bir simülasyon oluşturmaktır. Simülasyon oluştururken, ekosisteme ilişkin belirli parametrelerin yer aldığı, çeşitli algoritmalar ve kodlar ile oluşturulan bir model kullanıldı.

Bu çalışmada iki farklı simülasyon oluşturuldu. İlk simülasyon çalışmasında örnek bir canlı türü (Maratus volans) belirlendi ve simülasyona temel olacak bazı anahtar özellikler ve davranışlar seçildi. İkinci simülasyonda ise canlı olarak, tanımlanmış dinamikleri değiştirilebilir bir kurgusal herbivore (otçul) canlı ve besin/habitat olarak da bir bitki belirlendi.

Sonuç olarak, nesiller boyu gen aktarımı, doğal seçilim, canlılar arası etkileşim ve ekosistemin etkileri gibi popülasyon biyolojisi ile ilgili dinamikler gözlemlendi.

Bu projenin, girilen parametrelerle doğal ortamı bilgisayar simülasyonu üzerinde düzenlenebilir hale getirmek; çok maliyetli ve çok uzun sürecek veya düzenlenmesi imkânsız deney ve çalışmalar tasarlayan bilim insanları ve araştırmacılar için büyük bir kolaylık sağlamak ve en önemlisi de pedagojik anlamda eğitimlerde kullanılmak gibi önemli kazanımları olacağı düşünülmektedir.



**Anahtar kelimeler:** Simulasyon, Popülasyon, Biyoinformatik, Gen Aktarımı


## Amaç

Araştırma konusu yapılan canlıların, farklı koşul ve durumlarda nasıl tepkiler vereceğini ve popülasyonlarının çeşitli şartlarda nasıl değişimler göstereceğini gerçek hayatta gözlemlemek çok zor hatta neredeyse imkânsız ve çok maliyetlidir. Bu zorlu ve kısıtlı durumdan yola çıkarak, özellikle bilimsel çalışmalarda kullanılması amacıyla bu proje geliştirilmiştir. Bu proje, canlıların nasıl yaşadıklarını, nasıl davranışlar gösterdiklerini ve çeşitli durumlarda popülasyonlarında nasıl değişimler olduğunu bilgisayar ortamında gözlemleyebilmemizi sağlayan bir simülasyon çalışmasıdır. Çalışma ile amaçlanan canlı popülasyonunun temel dinamiklerini anlayabilmek ve ekosistem içerisindeki değişimlerini simüle ederek bilimsel çalışmalar için gözlem ortamı oluşturmaktır.

Bu projenin en temel faydalarını belirtmek gerekirse; sadece girilen parametrelerle aynı ortamı bilgisayar simülasyonu üzerinde düzenlenebilir hale getirmek, çok maliyetli, çok uzun sürecek veya düzenlenmesi imkânsız deney ve çalışmalar tasarlayan bütün bilim insanları ve araştırmacılar için inanılmaz büyük bir kolaylık sağlamak ve en önemlisi de pedagojik anlamda eğitimlerde kullanılabilir olmasıdır.

Ayrıca, bu simülasyon ile popülasyon biyolojisi ile ilgili karmaşık sorunların açıklaması çok daha kolay ve pratik bir şekilde anlatılabilecektir.

## Giriş

Canlılar da fiziksel dünyanın bir parçası olduğu için fizik yasalarına tabidir, ancak canlılığın en basiti dahi oldukça karmaşık özellikleri içerdiğinden biyolojide hatta canlı davranışlarının, dinamiklerinin konusu olan iktisat, tıp, psikoloji gibi diğer bilimlerde mekanizmalar, teoriler gibi açıklama biçimleri bulunsa da yerçekimi yasasına benzer doğa yasaları bulmak henüz mümkün olmamıştır. Bu problemin çözümü için çeşitli matematiksel veya biyolojik modeller oluşturulmuştur. Örneğin popülasyon dinamiklerinin anlaşılması için kullanılan Mendel prensipleri, Hardy-Weinberg hipotezi, Av-Avcı ilişkisi gibi modellerle biyolojik sistemler anlaşılmaya, açıklanmaya çalışılmaktadır.

Evrendeki çok az şey canlı sistemler kadar ilham verici ve gizem doludur. Örneğin, "hücre koleksiyonları insan aklının karmaşık, anlamlı düşünceler düşünmesine nasıl izin verebilir?" veya "Bir ekosistemdeki çeşitli organik ve inorganik oyuncular nasıl hareket ederler? Uzun vadeli istikrar üretmek için nasıl etkileşime giriyorlar?" veya "Embriyolar giderek karmaşık ve zarif biçimlerini nasıl elde ediyorlar?" Bunlar derin gizemlerdir. Araştırmacılar ve biyologlar bu tür soruları artan titizlikle ele almaya çalıştıkça, bu sistemlerde meydana gelen karmaşık etkileşimler hakkında bilgi edinmelerini sağlayacak araçlara ihtiyaç duyuyorlar ve şu anda mevcut olan en iyi araçlardan biri bilgisayar simulasyonlarıdır. (1).

1970'lerin başında Paulien Hogeweg ve Ben Hesper "biyotik sistemlerde bilişim süreçlerinin incelenmesi" anlamında "Biyoinformatik" terimini kullanmaya başladılar (2). Bugün interdisipliner bir bilim dalı olan biyoinformatik, biyolojik verileri, sistemleri anlamak, analiz etmek ve yorumlamak için yazılım, matematik, biyoloji, lojik, istatistik, mühendislik gibi birçok bilim dalını birleştirerek yöntemler ve yazılım araçları geliştiren disiplinler arası bir alandır. 2000'li yıllardan sonra ise biyoinformatik çalışmaları ağırlıklı olarak genetik ve moleküler biyoloji alanlarına yoğunlaşmıştır. Ülkemizde de TÜBİTAK desteği ile ulusal biyoinformatik ağı oluşturulması amacıyla çeşitli projeler geliştirilmektedir. Ayrıca ülkemizde bazı üniversitelerde yakın zamanda konu ile ilgili yüksek lisans ve doktora programları da açılmıştır.

Çalışma projemle ilgili literatür araştırması yaparken önce Türkçe akademik çalışmalar tarandı. TÜBİTAK ULAKBİM çatısı altında faaliyet gösteren akademik dergi ve elektronik makale platformu http://dergipark.gov.tr/ ve TÜBİTAK ULAKBİM veri tabanı https://trdizin.gov.tr/ de projeme benzer veya konuyla yakın ilişkili bir çalışma tespit edilemedi. Türkiye'de, bilgisayar simülasyon modelleme çalışmalarının, genellikle üretim süreçleri, organizasyon, afet, deprem, yangın gibi kriz durumları, iklim simülasyonları, eğitim gibi konularda yapıldığı tespit edildi. Biyoloji konusunda ise çok az sayıda çalışma mevcut olup yayınlanmış olanlar da özellikle genetik ve moleküler biyoloji konusundadır. Yükseköğretim Kurulu yüksek lisans ve doktora tezleri arşivi olan YÖK Ulusal Tez Merkezi tarandığında konuyla ilgili sadece iki yüksek lisans tezi ayrıca çalışma konumla ilişkili bir Türkçe konferans sunum metni tespit edildi. (3), (4), (5).

İngilizce ve Fransızca yaptığım bilimsel literatür taramasında ise çok sayıda ciddi çalışma tespit ettim. Tüm 525 geni, gen ürünleri ve etkileşimleri de dâhil olmak üzere Mycoplasma genitalium bakterisi için bir bilgisayar modeli oluşturuldu. (6).

Dünyanın birçok yerinde tarım için uzun süredir ciddi bir tehdit oluşturan böcek zararlıları, özellikle, çekirgelerin (Schistocerca Gregaria) simülasyonu ile ilgili bir çalışmada bir keçiboynuzu yetiştirme alanının mekânsal ve zamansal bir rüzgâr, sıcaklık, bağıl nem ve yağış gibi bilgileri de içeren bir simülasyon çalışması yapılmış, çekirge popülasyon dinamikleri anlaşılmaya çalışılmıştır. (7).

Başka bir çalışmada, Boophilus microplus (Canestrini) ve B. annulatus (Say) adlı kene türlerinin popülasyon dinamiklerinin simülasyonu için kapsamlı bir bilgisayar modeli geliştirilmiştir. Model, Avustralya'daki 7 lokasyonda ve Amerika'daki 23 lokasyonda tarihi hava durumu kullanıldığında, ilk nüfus artış hızı, üretim zamanı ve 3 yıllık nüfus yoğunluğu için kabul edilebilir değerler üretmiştir. Bu simülasyon modeli, Boophilus keneleri ile Babesia hastalığının bulaşının araştırılması için bir çerçeve sağlamıştır ve kontrol teknolojilerinin etkilerini incelemek ve Boophilus keneleri için daha verimli ve çevresel olarak kabul edilebilir eradikasyon stratejileri geliştirmek için kullanılabileceği öne sürülmüştür. (8).

Alman Federal Eğitim ve Araştırma Bakanlığı (BMBF) tarafından finanse edilen Sanal Karaciğer projesi ise 41 bilim ve sanayi kurumu ve Almanya'ya dağılmış 70 araştırma grubu tarafından yürütülen 43 milyon avroluk bir araştırma programıdır. Amaç, insan karaciğerinin fizyolojisini, morfolojisini ve işlevlerini temsil eden dinamik bir mantıksal tasarım, matematik model olan bir karaciğer simülasyonu üretmektir. Biyomoleküler ve biyokimyasal süreçlerden tüm organın anatomisine kadar tam bir organın gerçekten çok ölçekli bir bilgisayar modelini oluşturmayı amaçlayan dünya çapında ilk projedir. Ortaya çıkacak simülasyonun, vücudun önemli metabolik organı olarak tanımlanan karaciğerin daha iyi anlaşılmasına ve işlevlerinin hastalıklardan nasıl etkilendiğine katkıda bulunarak yeni tedaviler bulunmasına yardımcı olacağı düşünülmektedir. (9).

Bu proje temel olarak bir simülasyon çalışmasıdır. Simülasyon oluştururken, öncelikle gerçek dünyadaki bir **sistemden** bazı detayları atıp özellikle ilgilendiğimiz dinamiklerle ilgili karmaşık ve ayrıntılı bir gerçekliği temsil etmesi için **özet** çıkarırız. Bu özet bizim bilimsel olarak ilgilendiğimiz sistemin araştıracağımız hususiyetlerini içeren bir **modelidir**. Bu model, mantıksal tasarım, algoritma, analiz ve bir yazılım dili yardımıyla bir **simülasyona** dönüştürülür. Bu simülasyon bir bilgisayar programı formundadır. Bu simülasyon ve analizin amacı bilimsel tahmin ve öngörülerde bulunmak ve araştırdığımız sistemin nasıl, hangi mekanizmalarla çalıştığının anlaşılmasına yardımcı olmaktır. (10).

Bu simülasyon çalışmada öncelikle örnek bir canlı türü belirlendi. (Maratus volans) Daha sonra simülasyonda ele alınacak temel konular, Maratus volans ve otçul canlılar ve Ortam/Besin olarak Çim ile ilgili geçerli kaynak bilgileri araştırıldı. Sonra simülasyona temel olacak bazı anahtar özellikler ve davranışlar seçildi ve Maratus volans için bir simulasyon oluşturuldu. Daha sonra tanımlanmış dinamikleri değiştirilebilir bir kurgusal otçul (herbivore) canlı ve Besin/Habitat olarak bir bitki simülasyonu tasarlandı. Bu ikinci simülasyonda model otçul canlıya hareket, besin arama gibi farklı özellikler ve simülasyona zaman faktörü de eklendi ayrıca simülasyonun grafik, görsel animasyon düzeyi de daha sofistike hale getirildi.

İlk simülasyonda, bu canlı türünün (Maratus volans) tanımlı özellikleri ile popülasyonunun davranışları ve değişimi ile ilgili gözlemler yapılabilir. Doğal seleksiyon

hakkında fikir edinilebilir, popülasyon artış hızını görülebilir ve popülasyon içindeki genetik aktarım hakkında gözlemler yapılabilir.

İkinci simülasyonda, ortam/besin, zaman faktörü, kurgusal iki canlı türü ve canlı davranışlarıyla ilgili daha karmaşık özellikler eklenerek, temsil düzeyi daha gerçekçi hale getirildi. Bu sayede daha kompleks bir simülasyon içinde aynı kavramlar gözlenerek, anlaşılabilir.

Bu simülasyon modellemesi ile gerçek dünyadaki hala tam anlaşılamamış bir konunun hızlı, güvenli, verimli bir çözümü için etkili bir araç geliştirilmiştir ayrıca kolayca gözlemlenen ve anlaşılan önemli bir analiz yöntemi sağlanmıştır. Gerçek bir popülasyon veya ekosistem üzerinde deneyler yapmak maliyet veya zaman nedeniyle neredeyse imkânsızdır veya pratik değildir. Yaptığım bu çalışma, canlı davranışları, genetik aktarım ve popülasyon dinamikleri ile ilgili dijital bir ortam üzerinde deney yapılmasına imkân sağlayabilir. Özellikle ikinci simülasyonda animasyonun zenginleştirilmesi ile çalışma, daha açıklayıcı ve pedagojik bir araç olarak hizmet eder, bilgi durumunu daha açık bir şekilde temsil eder.

## Yöntem

Öncelikle örnek bir canlı türü seçildi. Avusturya tavus kuşu örümceği (Maratus volans) Canlının kendisi ve davranışları bilimsel literatürden araştırıldı ve bu canlıyı kullanarak bir simülasyon örneği oluşturuldu.

Yazılım dili olarak JavaScript, html ve Css, üçü bir arada kullanarak web tabanlı interaktif bir sistem oluşturulması hedeflendi. Böylece tüm dünya tarafından erişimi sağlanabilir olması amaçlandı.

Program kodu, Nesne yönelimli programlama (Object-oriented programming) yöntemi kullanılarak yazıldı. Nesne yönelimli programlama, bütün dünyayı nesneler (objects) ve nesneler arası ilişki şeklinde gören bir model kullanılmasına dayalı bir metottur. Bu programlama yöntemine göre her nesne bir sınıfa (class) aittir ve bu sınıftan türetilir. (11). Proje için bu metodun kullanılması daha uygun görüldü.

Maratus volans'ın gerçek özellikleri içinden;

- Dişi/Erkek
- Anne/Baba olma (Ebeveyn)
- Üreme isteği
- Üreme davranışı
- Güç seviyesi
- Dans yeteneği
- Ölüm (Belirlenen ömürle doğal veya dişi tarafından yenme suretiyle ölüm)

hususiyetleri seçildi. Simülasyon kodları yazılıp çalıştırıldıktan sonra gözlemlenmek istenen hususların bağımlı/bağımsız değişkenlere ilişkin grafiklerle daha iyi anlaşılması için grafik kodları yazıldı ve çalıştırıldı. Pratik ve estetik olması nedeniyle grafik kodları chart.js adındaki açık kaynak kodlu kütüphane yardımıyla hazırlandı. Görülmesi istenen verileri girerek değişimleri saptayabildiğimiz grafikler koda eklenmiş oldu.

Bu yapılan ilk simülasyon bir örnek ve temel oluşturdu. Daha sonra çalışma daha sofistike hale getirilmek istendi ve bir ortam içinde iki canlıdan oluşan animatif bir

simülasyon tasarlandı. Bir otçul canlı ve ortam/besin olarak çimen seçildi. Bunların özellikleri araştırıldı. Otçul canlı, dişi/erkek olarak, cinsiyet ve renk dışında diğer genotipik ve fenotipik özellikler açısından farklı olarak tanımlanmadı.

Otçul canlıya, 1. modeldeki hususiyetlere ek olarak;

- Açlık seviyesi (60+ ise Besin arıyor, 100+güç üzerinde ise açlık nedeniyle ölüm)
- Besin arama
- Pozisyon x-y koordinatı
- Yön-Hareket (8 yöne hareket edebilme. Bunları kendi seçiyor ve seçtiği yöne doğru ilerleyebilir)
- Zekâ (0-30)
- Çekicilik seviyesi ((Güç*0.4 + Zekâ x15)*100/490)
- Hissetme alanı (yarıçap olarak hesaplanıyor=> 5+Zekâ'nın ⅓'ü) (Hissetme alanını hayalî bir daire gibi düşünebiliriz. Canlı sadece bu alan içindeki canlılarla etkileşime girebiliyor. Bu dairenin içindeki canlılar aynı zamanda potansiyel partner olabilecek)
- Doğum oranı (Bir batında 1 ila 14 yavru doğurma olasılığı)
- Rengi (Erkekse mavi, dişi ise pembe, üreme esnasında sarı)

özellikleri eklendi. Ayrıca beslenme hususiyeti eklendiği için ölüm özelliği de buna uygun detaylandırıldı. (Belirlenen ömürle doğal veya Açlık 100+canlının gücü üzeri ise ölüm) Bunun dışında, 1. modeldeki diğer mevcut hususiyetler de temsil düzeyini artırmak amacıyla canlı birey ve popülasyon dinamikleri açısından da daha kompleks hale getirilmeye çalışıldı. Örneğin Üreme faaliyeti ile Yaş arasında bir ilişki düzenlendi. Otçul canlının üreyebilmesi için belli bir yaşa gelmesi gerekiyor. En önemlisi popülasyon içerisindeki 1.derece akrabalarla (Anne/Baba/Yavru) çiftleşemiyor.

Bitkilerden oluşan bir zemin olarak çim seçildi. Çim, hem kendisi ikinci bir canlı hem otçul canlının besini hem de habitat/ortam olarak tasarlandı. Çim için;

- Büyüme hızı
- Maksimum büyüme seviyesi

gibi özellikler seçildi. Çim, besin dışında ortam, çevre özelliği gösterdiği için simülasyonda daha net gözükmesi için renk kodları detaylandırıldı. Zemin varsayılan olarak toplam 300 kareden oluşturuldu. İstenildiği takdirde çok kolay bir şekilde bu karelerin boyutu ayarlanabilir, istenildiği kadar arttırılıp azaltılabilir şekilde düzenlendi.

Böylece canlıları bir animasyon içinde, yön seçip hareket etme gibi daha karmaşık özelliklerinin temsiliyle, kendi kendilerine yaşamalarını göstermenin ve gözlemlemenin mümkün olabileceği, zaman, ortam/besin gibi daha farklı etmenlerin bulunduğu daha kompleks bir simülasyon hazırlanıp, çalıştırıldı. Yine araştırma, gözlemleme konusu olabilecek istenen bazı hususların, bağımlı/bağımsız değişkenlere ilişkin grafiklerle daha iyi anlaşılması için grafik kodları yazıldı ve çalıştırıldı. Grafikler için yine chart.js kütüphanesi tercih edildi. Böylece değişkenlerin zaman içinde gösterdikleri artışlar görsel bir şekilde ifade edildi.

**Proje İş-Zaman Çizelgesi**

| AYLAR | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **İşin Tanımı** | **Nisan** | **Mayıs** | **Haziran** | **Temmuz** | **Ağustos** | **Eylül** | **Ekim** | **Kasım** | **Aralık** | **Ocak** |
| **Literatür Taraması** | | | | **X** | **X** | **X** | **X** | **X** | **X** | |
| **Arazi Çalışması** | | | | | | | | | | |
| **Verilerin Toplanması ve Analizi** | | | | | | **X** | **X** | **X** | **X** | |
| **Proje Raporu** | | | | | | | | | **X** | **X** |

## Bulgular ve Gerçekleşme

Avusturya Tavus kuşu örümceği, Maratus volans, davranışları ve özellikleriyle gerçekten çok ilginç bir canlıdır. Uzunluğu 3-5 milimetre arasında değişen küçük bir örümcek türüdür ve güney Avustralya bölgesinde bulunur. Resim 1 ve 2'de görüldüğü gibi erkekler, dişilerinden kuyruklarıyla ayırt edilebilir, dişilerin kuyrukları olmaz.(12) Kendi boyutlarına göre 20 kat daha yukarı zıplayabilirler. Latince ismindeki 'volans' uçmak anlamına gelir, Fransızca'da da yine aynı kökten gelen 'voler' uçmak fiilidir ve taksonomide ad verilirken örümceğin bu kadar yukarı zıplayabilmelerinden ötürü bu isimlendirme yapılmıştır. Maratus volans'ın belki de en dikkat çeken özelliği ise üreme safhasıdır. Erkekleri, dişilerini etkilemek için üreme öncesinde bir dans gösterirler ve dişi yeterince etkilenmez ise dişi erkeği yer; etkilenirse çiftleşirler ve tür neslini devam ettirmiş olur. Etkileyemez ise dişi tarafından yenilecek olan erkek örümceğin zıplayarak kaçma ihtimali de vardır. (13), (14), (15).

Resim 1. (12)

Resim 2. (12)

Maratus volans simülasyonunda canlının temel bazı özellikleri kullanılmaya dikkat edildi. Simülasyonda davranış olarak gerçek hayatta olduğu gibi erkekler öncelikle bir partner buluyor. Üreme faaliyeti için erkeklerin dans yeteneği, dişilerin üreme isteği ile eşit ya da ondan büyük olmalıdır. Eğer buldukları partnerlerin yani dişilerin üreme isteği, erkeklerin dans yeteneğinden fazlaysa, erkeğin ve dişinin güç seviyesine göre erkek ya kaçıyor ya da ölüyor.

Simülasyonda, dans yeteneği, üreme isteği, renk ve kuyruk görseli gibi özelliklerle Dişi/Erkek ayırt edilebiliyor. Üreme faaliyeti ve neticesinde popülasyondaki nesiller içerisindeki gen aktarımını gözlemleyebiliyoruz. Popülasyon içinde Anne/Baba (Ebeveyn) kavramları temsil edilebiliyor. Belirlenen ömürle doğal veya dişi tarafından yenme suretiyle ölümü gözlemleyebiliyor, popülasyona etkisini izleyebiliyoruz. Jenerasyon arttıkça hem popülasyon nüfusu artıyor hem de dans yeteneği değeri yüksek erkek bireylerin popülasyon içinde ayakta kaldığını (doğal seçilim) görebiliyoruz.

İkinci simülasyonda ortam/besin olarak modellenen Çim'in 0, I, II, III, IV olmak üzere 4 farklı durumu/evresi görülebiliyor.

- **0. Evre** Ortamın, ekosistemin en kuru hali. Yenilebilecek, tüketilebilecek bir çimin olmadığı tamamen kuru toprak
- **1. Evre** biraz büyümüş, tüketilebilir hali
- **2. Evre** daha da büyümüş yine bir kısmı tüketilmiş durumu
- **3. Evre** Maksimum en büyümüş ve olgunlaşmış halini gözlemleyebiliyoruz

Üstünde bulunan otçul canlılar, gelişmiş ise çimi yiyebiliyorlar ve tüketildiği zaman çim, geni için kodla tanımlanmış parametrelere göre hızlı veya yavaş büyüme gösteriyor.

Bu ekosistemi tamamlayan otçul canlının dişi/erkek olarak popülasyon içindeki başlangıç nüfusları rastgele (random) olarak belirleniyor. Otçul canlı, 8 farklı yöne hareket edebiliyor, bunları kendi seçiyor ve seçtiği yöne doğru ilerleyebiliyor. Açlık seviyesine göre

(60’dan yukarı ise ) besin arıyor. Besin bulunca yiyor, bulamaz ise ve açlık seviyesi 100+güç seviyesinin üzerine çıkarsa doğal ölüm dışında besin eksikliğinden ölüyor. Her canlının His yarıçapı (5 + Zekâ’nın ⅓’ü) diye tanımladığımız bir değişkeni var ve otçul canlıya, yarıçapı bu değişkeni ifade eden dairesel bir **görüş alanı** sağlıyor. Bu görüş alanındaki diğer canlılarla etkileşime girebiliyor. Popülasyon içinde, erkek canlıların görüş alanına giren dişiler, otçulun potansiyel partnerleri olabiliyor. Erkek bireylerin çekicilik seviyesi ((Güç*0.4 + Zekâ x15)*100/490) üreme olasılığını belirliyor. Dişi,

- Erkek canlının görüş alanında ise,
- Eski partneri değilse ve
- 1.derece akrabası (Anne/Baba ve Yavru) değilse,

çekicilik seviyesinin belirlediği ihtimale göre üreme faaliyetinin gerçekleştiğini gözlemleyebiliyoruz. Doğum başına yavru sayısı bir batında 1 ile 14 arasındadır (tavşanlar örnek alınarak hesaplandı). Çekicilik seviyesi, görüş alanı gibi hususiyetleri belirleyen Zekâ (ilk başta 0-20 daha sonra ya gen aktarımı ile ya da 0-30) simülasyonun en başında rastgele olarak belirleniyor ancak popülasyon içindeki sonraki jenerasyonlara % 80 (istenildiği gibi değiştirilebilir) ihtimalle genetik olarak aktarılıyor.

Gözlemler birçok parametre üzerinde yapılabiliyor. Bu projede ortamın popülasyona etkisini rahatlıkla görebiliyoruz. Çim sayısı azaldıkça popülasyondaki artış da azalıyor. Çimin büyüme hızı da keza popülasyon artışını doğru orantılı olarak etkiliyor. Ayrıca canlıların etkileyicilik seviyeleri, zeka ve güçleri ile doğru orantılı. (Ne kadar zeki ve güçlü olursa dişi o kadar etkileniyor.) Bunun sonucunda daha zeki ve güçlü olanların diğerlerine göre neslini devam ettirme olasılığının arttığını tahmin edebiliriz. Simülasyonda da tahminimiz doğrultusunda zaman geçtikçe zekâ ve güç ortalamalarının arttığını görebiliyoruz.

**Proje Kodlarının Açıklanması**

Kodlar ilk hazırlandığında yazılım dillerinin İngilizce olması nedeniyle İngilizce hazırlanmıştı. Ancak yarışmaya katılmadan önce değişkenlerin bir çoğu daha kolay anlaşılması için Türkçeleştirildi. Ancak bir çok yazılım terimi ve bazı İngilizce terimler Türkçede karşılığı bulunmadığından kodun bazı kısımları İngilizce olarak bırakıldı.

Kodların açıklamaları fotoğraflar üzerinde yorum blokları halinde yeşil renkteki yazılar ile belirtildi.

## 2. Simülasyon:

*Ana kod:*

```
const ekran = document.getElementById("ekran"); //simülyasonun <canvas> 'ı ekran değişkenine atandı.
const ctx = ekran.getContext("2d");//ekranın context'i ctx değişkenine atandı.
const cimKU = 40; //her bir çim karesinin bir kenarının uzunluğu atandı.(*)
let t; //t adında bir değişken tanımlandı
const zemin = [];//zemin adında bir dizi tanımlandı
const canlilar = [];//canlilar adında bir dizi tanımlandı
let ilkD; //ilkD adında bir değişken tanımlandı.
const imlec = {
    x:0,
    y:0
};//imlec adında bir nesne tanımlandı
let populasyon = 80; //başlangıç populasyonu girildi.(*)

const setup = () => {
    ilkD = new Date();//ilkD değişkenine kod çalıştırldığı andaki tarih atandı.

    for(let i = 0; i < ekran.width/cimKU; i++){
        for(let j = 0; j < ekran.height/cimKU; j++){
            zemin.push(new Tile(i*cimKU,j*cimKU));
        }
    }//bu kod ekranın boyutuna göre sığacak tüm çimleri zemin dizisinin içinde ittirdi.

    for (let i = 0; i < populasyon; i++){
        canlilar.push(new Canli());
    }//bu kod canlılar dizisine başlangıç popülasyonu kadar yeni Canlı nesnesi ittirdi.
    console.log("Setup done."); //Bu kod konsola hazırlık sürecinin sorunsuz bir şekilde bittiğini yazdırdı.
}
let loop = () => {
    ctx.clearRect(0,0,ekran.width,ekran.heigth);//Bu kod döngü her döndüğünde ekranı tamamen temizledi.
    allFuncs.runForArr(zemin,"draw"); //bu kod bütün zemin dizisindeki elemanların "draw" adındaki fonksiyonunu çalıştırdı
    allFuncs.runForArr(canlilar,"show"); //bu kod bütün canlilar dizisindeki elemanların "show" adındaki fonksiyonunu çalıştırdı
    t = requestAnimationFrame(loop) //bu kod döngüyü devam ettirirken t değişkenine döngünün son halini atadı.
}
setup(); //bu kod setup fonkiyonunu çağırdı.(14-28. satırlar arası)
requestAnimationFrame(loop); //bu kod döngüyü başlattı (29-34. satırlar arası)
onmousemove = (e) => {imlec.x = e.clientX-8;imlec.y = e.clientY-8};//bu kod her fare oynadığında imlecin x ve y koordinatını aldı.
```

*Çimen 1:*

```
class Tile{// Tile adında bir class oluşturuldu
    constructor(x,y){
        this.x = x; //x kordinatı parametre olarak giriliyor.
        this.y = y; //y kordinatı parametre olarak giriliyor.
        this.w = cimKU; //genişliği cimKU değişkeninin değeri olarak girildi
        this.h = cimKU; //yüksekliği cimKU değişkeninin değeri olarak girildi
        this.status = 3; // status adında bir değişken atandı (Maksimum 3 minimum 0 oluyor.)
        this.r = 0; //renk kodları için rgb'nin r kısmı yani kırmızılık miktarı atandı
        this.g = 153; //renk kodları için rgb'nin g kısmı yani yeşillik miktarı atandı
        this.growthPerc = 1; //her bir status arasındaki büyüme seviyesi olarak growthPerc adında bir değişken tanımlandı
        this.growthRate = 0.2; //büyüme hızı tanımlandı.(*)
        this.isGrown = allFuncs.toInt(allFuncs.randomFloat(50,70)); //her bir status tamamlaması için gereken büyüme seviyesi rastgele (50-70 arası) hesaplandı.
    }

    draw(){
        this.grow();//çim yaşarken ilk önce büyüyor (bkz. 39-48 satırlar arası)
        this.controlStatus(); //çim daha sonra status değerine göre rengini belirliyor (bkz. 24-38.satırlar arası)
        ctx.fillStyle = `rgb(${this.r},${this.g},0)`;
        ctx.fillRect(this.x,this.y,this.w,this.h);
        ctx.fillStyle = "#000000";
        //(18-20. satırlarda) çim ekrana çiziliyor.
    }
    controlStatus(){
        if (this.status == 3){
            this.r = 0;
            this.g = 153;
        }else if (this.status == 2){
            this.r = 76;
            this.g = 153;
        }else if (this.status == 1){
            this.r = 153;
            this.g = 153;
        }else if (this.status == 0){
            this.r = 153;
            this.g = 76;
        }
    }
```

## Çimen 2:

```
    }

    grow(){
        if (this.status < 3){
            if (this.growthPerc >= this.isGrown){
                this.status++;
                this.growthPerc = 1;
            }else {
                this.growthPerc += this.growthRate;
            }
        }
    }
    //bu fonksiyon status değeri maksimumdan düşmüş çimler için büyüme işlemini gerçekleştiriyor.


}
```

## Canli 1:

```
class Canli{ //Canli adında bir class oluşturuldu.
    constructor(){//constructor kısmındaki her değişken kullanıcı isteğine göre değiştirilmeye, oynanmaya müsait.
        this.cinsiyet = allFuncs.choice(["male","female"]); //cinsiyeti rastgele dişi veya erkek olarak belirleniyor.
        this.karsiCins = this.cinsiyet=="male" ? "female" : "male"; //karşı cinsi cinsiyetinin tam tersi olarak belirleniyor.
        this.renk = this.cinsiyet =="male" ? "rgb(40,80,250)" : "rgb(255,40,160)" //rengi cinsiyetine göre belirleniyor.
        this.aclık = 0; //açlık seviyesi atanıyor.

        this.pos = new allFuncs.Vektor(allFuncs.randomFloat(1,ekran.width-1),allFuncs.randomFloat(1,ekran.height-1));
        //pozisyonu yeni bir Vektor nesnesi olarak belirleniyor.

        this.anne = 0; //Annesi olmadığı için 0 olarak atanıyor.
        this.baba = 0; //Babası olmadığı için 0 olarak atanıyor.
        this.cocuklar = []; //cocuklar adında bir dizi atanıyor
        this.DNA = { //dna adında bir nesne belirleniyor(Bu nesnedeki değişkenler nesiller boyu aktarılan özellikler)
            guc: allFuncs.toInt(allFuncs.randomFloat(0,100+1)), // Rastgele 0-100 arası bir güç değeri atanıyor.
            intelligence: allFuncs.toInt(allFuncs.randomFloat(0,20+1)), // 0-20 arası bir zeka değeri atanıyor. (İlk başlayanlar 0-20 diğerleri 0-30).
            charm: undefined, //altta belirlenecek (scope kuralları yüzünden)
            visionRadius: undefined,// 17. satır ile aynı sebep
        };
        this.partner = 0 //Partneri olmadığı için 0 olarak atanıyor.
        this.DNA.charm = allFuncs.percit(this.DNA.guc+this.DNA.intelligence*1.2,0,136);
        //canlinin gücü + zekası*1.2 hesaplanıyor. Bu değer 0 ile 136 arasında yüzde olarak hesaplanıyor.(21.satır).
        //Örnek 50+15*1.2 = 68, 68 sayısı 136 nın yarısı bu yüzden charm değeri 50 olarak atanıyor.(21.satır)

        this.DNA.visionRadius = 30 + this.DNA.intelligence/3; //görüş yarı çapı hesaplanıyor
        this.dir = this.createdir(); //(bkz. 33-37. satırlar)
        this.dirCounter = 0; //bir yöne gittiği süre boyunca artan sayaç değeri atanıyor.
        this.oldPartners = [] //oldPartners adında bir dizi atanıyor.
        this.vitalActivity = true; //yaşamsal faaliyetleri kontrol eden "boolean" atanıyor.
        this.age = 0; //yaşı 0 olarak başlıyor.
        this.lifespan = allFuncs.toInt(allFuncs.randomFloat(120,150)) * 5 //ömrü hesaplanıyor.
    }
    createdir(){
        let p = new allFuncs.Vektor(allFuncs.choice([-1,0,1]),allFuncs.choice([-1,0,1])); //Rastgele olarak 9 farklı yönden biri seçiliyor
        p.x==0&&p.y==0 ? p.x=allFuncs.choice([-1,1]) : null; //Eğer yön (0,0) olarak seçilirse yani hareketsiz kalma x eksenine doğru rastgele bir yön yeniden seçiliyor.
        return p //bu yön nesnesini döndürüyor.
    }
```

## Canli 2:

```
    show(){
        //Ekranın içinde kal
        this.pos.x = allFuncs.limit(this.pos.x,0+1,ekran.width-11);
        this.pos.y = allFuncs.limit(this.pos.y,0+1,ekran.height-11);

        //Görsel kısım
        ctx.beginPath();
        ctx.arc(this.pos.x,this.pos.y,5,0,2*Math.PI)
        if (this.vitalActivity){
        ctx.fillStyle = this.renk;
        }else {
        ctx.fillStyle = "rgb(255,255,0)";
        }
        ctx.fill();
        ctx.stroke();
        //ctx.fillRect(this.pos.x,this.pos.y,10,10);


        this.vision();

        //Genel davranışlar
        if(this.vitalActivity){
        this.behaviours();
        this.aclık+=0.4;
        this.age+=0.025;
        }
        //Ölüm kontrolu
        ((this.age > this.lifespan) || (this.aclık > 100+this.DNA.guc) ) && canlilar.splice(canlilar.findIndex(el => el==this),1) //canlı ölürse dünyadan siliniyor.

    }
```

## *Canli 3:*

```
    behaviours(){
        (this.aclık > 60) && this.searchForfood(); //açlık değeri 60'tan fazlaysa yemek aramaya başlıyor.
        (this.age > 15) && (this.cinsiyet=="male") && (this.partner == 0) && this.lookForPartner();
        //yaşı 15'ten büyükse ve erkekse ve partneri yoksa partner aramaya başlıyor.(71. satır)
    }
    searchForfood(){
let foodTile = zemin[zemin.findIndex(elem => elem.x == allFuncs.closestMultiple(this.pos.x + 5,cimKU,-1) && elem.y== allFuncs.closestMultiple(this.pos.y + 5,cimKU,-1))];
//Üstünde bulunduğu çimi bir algoritmayla hesaplıyor. (75. satır)
            if (foodTile.status < 1){
                //Eğer üstünde bulunduğu çim yenemez durumdaysa kendine bir yön seçiyor. Ve belirli bir süre o yönde ilerliyor.
                if (this.dirCounter % 15 == 0){
                    this.dir = this.createdir();
                }
                this.pos.ekle(this.dir)
                this.dirCounter++;
            }else{
                //Eğer üstünde bulunduğu çim yenebilir durumdaysa bu sefer çimi yiyor.
            this.eat(foodTile)
            this.dirCounter = 0;
            }
    }

    eat(tile){
        this.aclık = 0;
        tile.status--;
    }//Bu kod ile çimi yediği zaman açlığı 0'a düşüyor ve çimin statusu azalıyor.

    vision(){
        //Etkileşime girdiği alanı canlının üzerine geldiğimizde göstermesini sağlayan fonksiyon.
        if (allFuncs.uzaklik(imlec.x,imlec.y,this.pos.x,this.pos.y) < 5 || allFuncs.uzaklik(imlec.x,imlec.y,this.pos.x+5,this.pos.y+5) < 5){
        ctx.beginPath();
        ctx.globalAlpha = 0.4;
        ctx.arc(this.pos.x+5,this.pos.y+5,this.DNA.visionRadius,0,2*Math.PI);
        ctx.fillStyle = 'rgb(80,80,80)';
        ctx.fill();
```

## *Canli 4:*

```
        ctx.lineWidth = 0.6;
        ctx.globalAlpha = 1;
        ctx.stroke();
    }
    }

    lookForPartner(){
        //bu fonksiyon görüş alanındaki tüm canlıları alıp içlerinden belirli şartları karşılıyan bir canlıyı seçip onunla partner olmasını sağlıyor.
        let mousesInRange = []
        for(let i = 0; i < canlilar.length;i++){
            if (allFuncs.uzaklik(this.pos.x,this.pos.y,canlilar[i].pos.x,canlilar[i].pos.y) < this.DNA.visionRadius){
                mousesInRange.push(canlilar[i])
            }
        }
        for(let i = 0; i < mousesInRange.length;i++){
        if(mousesInRange[i].partner == 0 && mousesInRange[i].cinsiyet == this.karsiCins && !allFuncs.haveCommon(mousesInRange,this.oldPartners) && !allFuncs.haveCommon(mou
                this.partner = mousesInRange[i];
                mousesInRange[i].partner = this;
                //Daha sonra partneriyle charm seviyesi kadar ihtimalle ürüyor.(Dolayısıyla charm değeri ne kadar fazlaysa üreme ihtimalı o kadar fazla oluyor.)
                if(allFuncs.probability(this.DNA.charm-15,true,false)){ //çıkarılan sayı arttıkça üreme ihtmimali azalıyor. Bu da sadece çekici olanların nesillerini sürdü
                 //if(allFuncs.probability(allFuncs.percit(this.DNA.intelligence,0,30),true,false)){
                    this.reproduce();
```

## *Canli 5:*

```
                }else {
                 //Eğer üreyemezlerse ayrılıyorlar.
                 this.partner.oldPartners.push(this)
                 this.oldPartners.push(this.partner)
                 this.partner.partner = 0;
                 this.partner = 0;
                }
            }
        }
    }

    reproduce(){
        //Ürerken yaşamsal faaliyetler duruyor (Yemek aramıyorlar, acıkmıyorlar, ölmüyorlar)
        this.vitalActivity = false;
        this.partner.vitalActivity = false;
        //Partnerler yan yana geliyorlar.
        this.partner.pos.y = this.pos.y;
        //Çocuk yapıyorlar. Ve fp adındaki değişkene eski partnerin değeri atanıyor.
        let fp = this.makeChild();
        //2 saniye sonra yaşamaya devam ediyorlar.
    setTimeout(()=>{
        fp.vitalActivity = true;
        this.vitalActivity = true;
        },2000)
    }

    makeChild(){
        //Anne ve babanın genleriyle ihtimaller doğrultusunda gen aktarımı gerçekleştiyor. Ve 1-14 arasında yavru yapıyorlar.
        //Buradaki ihtimaller ve birçok parametre kullanıcı tarafından değiştirilebilir.
        for (let i = 0; i < allFuncs.toInt(allFuncs.randomFloat(1,14+1)); i++){
        let child = new Canli();
        child.pos = new allFuncs.Vektor(allFuncs.ortalama(this.pos.x,this.partner.pos.x),this.pos.y + 15);
```

## *Canli 6:*

```
        child.baba = this.cinsiyet=="male" ? this : this.partner;
        child.anne = this.cinsiyet=="female" ? this : this.partner;
        child.DNA.guc = allFuncs.probability(80,allFuncs.toInt(allFuncs.randomFloat(child.baba.DNA.guc-10,child.baba.DNA.guc+10)),allFuncs.toInt(allFuncs.randomFloat(0,10
        child.DNA.intelligence = allFuncs.probability(80,allFuncs.toInt(allFuncs.randomFloat(child.baba.DNA.intelligence-10,child.baba.DNA.intelligence+10)),allFuncs.toIn
        child.DNA.charm = allFuncs.percit(child.DNA.guc*0.4+child.DNA.intelligence*15,0,100*0.4+30*15);//Katsayılar değiştirilebilir. Çekicilik seviyesini ne kadar dolayl
        child.DNA.visionRadius = 30 + child.DNA.intelligence/3;

        //Yavru dünyaya ekleniyor.
        this.cocuklar.push(child);
        this.partner.cocuklar.push(child);
        canlilar.push(child)
        }

        //Üredikten sonra ayrılıyorlar.
        this.partner.oldPartners.push(this)
        this.oldPartners.push(this.partner)
        let pf = this.partner
        this.partner.partner = 0;
        this.partner = 0;
        return pf //eski partneri döndürüyor
    }
}
```

*Grafik 1:*

```
const ctx2 = document.getElementById('grafik1').getContext('2d'); //İlk grafiğin context'i alınıyor
const ctx3 = document.getElementById("grafik2").getContext("2d"); //İkinci grafiğin context'i alınıyor
const grafikZaman = [];

const paramortalama = (paramName) => {
    //Bu fonksiyon verilen parametrenin tüm canlılar arasındaki ortalamasını hesaplıyor
    let arr = []
    for (let i = 0; i < canlilar.length; i++){
        arr.push(eval("canlilar[i]." + paramName))
    }
    return allFuncs.ortalama(...arr)
}

var timePassed = 0; // Kodun çalışmasından itibaren geçen zaman değeri.
var sec = 0; // bu değişken, bu kod içinde zaman ile ilgili hesaplarda kullanılan bir değer.
var df = 1; // bu değişken, bu kod içinde zaman ile ilgili hesaplarda kullanılan bir değer.

//Zaman-Popülasyon grafiği
var timePopulationChart = new Chart(ctx2, {
type: 'line',
data: {
labels:[],
datasets: [{
    label: "Popülasyon",
    data:[],
    fill:false,
    backgroundColor: "rgba(255, 99, 132, 0.2)",
    borderColor: "rgba(255,99,132, 1)",
    borderWidth: 1
}]
},
options: {
responsive: false,
maintainAspectRatio: false
}
});
```

*Grafik 2:*

```
37  //Zaman-Zeka grafiği
38  var timeIntelligenceChart = new Chart(ctx3, {
39      type: 'line',
40      data: {
41      labels:[],
42      datasets: [{
43          label: "Zeka",
44          data:[],
45          fill:false,
46          backgroundColor: "rgba(150, 200, 150, 0.2)",
47          borderColor: "rgba(150, 200, 150, 1)",
48          borderWidth: 1
49      }]
50      },
51      options: {
52      responsive: false,
53      maintainAspectRatio: false
54      }
55  });
56
57
58  setInterval(()=>timePassed = Math.abs(Math.round((ilkD - new Date())/1000/30)),10);
59  //Bu kod her 10 saniyede bir kodun çalışmasından beri kaç 30 saniye geçtiğini hesaplıyor.( 58.satır)
60
61  //Grafiklerin sayfada yerleştirilmesi ve boyutlandırılması
62  document.getElementById("grafik1").style = "position: absolute; top: 360px; left: 820px;"
63  document.getElementById("grafik1").width = "400";
64  document.getElementById("grafik1").height = "400";
65
66  document.getElementById("grafik2").style = "position: absolute; top: 0px; left: 820px;"
67  document.getElementById("grafik2").width = "400";
68  document.getElementById("grafik2").height = "400";
```

*Grafik 3:*

```
69
70
71
72  var upChart = () => {//Bu fonksiyon grafikleri güncelliyor.
73      //(73,75,87,89. satırlar geçen zamanı hesaplıyor.)
74      !grafikZaman.includes("Dakika: " + timePassed.toString()) && grafikZaman.push("Dakika: " + timePassed.toString());
75      if (df == grafikZaman.length){//Bu kontrol sadece 30 saniyede bir çalıyor.
76      let secText= sec%1==0 ? "Dakika: " + sec.toString()+":00" : "Dakika: " + Math.floor(sec).toString()+":30"
77
78
79          timePopulationChart.data.labels.push(secText)//Gecen zaman grafiğe veri olarak atanıyor.
80          timePopulationChart.data.datasets[0].data.push(canlilar.length);//Popülasyon grafiğe veri olarak atanıyor.
81          timePopulationChart.update();//Grafik güncelleniyor.
82
83          timeIntelligenceChart.data.labels.push(secText)//Gecen zaman grafiğe veri olarak atanıyor.
84
85          timeIntelligenceChart.data.datasets[0].data.push(allFuncs.percit(paramortalama("DNA.intelligence"),0,30))
86          //Zeka ortalaması yüzdelik değer olarak grafiğe veri olarak atanıyor. (85.satır)
87
88          timeIntelligenceChart.update();//Grafik güncelleniyor.
89
90          sec+=0.50;
91
92          df++;
93      }
94      requestAnimationFrame(upChart); //güncelleme işlemi için döngü oluşturuluyor.
95  }
96
97  requestAnimationFrame(upChart); //döngü başlatılıyor.
```

## *Fonksiyonlar 1:*

```
//Kod içinde genel olarak kullanılan fonksiyonlar bu sayfada derlendi ve allFuncs adındaki nesnenin içinde depolandı.
//Bu nesne istenirse export edilip başka projelerde de bir modül olarak kullanılabilir.
const allFuncs = {
  noLoop: (anim) => {
    cancelAnimationFrame(anim);
    console.log("Loop ended.")
  },//Bu fonksiyon döngüleri durdurur.
  limit: (n, min, max) => Math.max(Math.min(n, max), min), //Bu fonksiyon girilen değeri yine girilen maksimum ve minimum değerleri arasında tutar ve döndürür.
  uzaklik: (x1,y1,x2,y2) => Math.sqrt(Math.pow(Math.abs(x1-x2),2) + Math.pow(Math.abs(y1-y2),2)), //Bu fonksiyon 2 boyutlu düzlemde 2 kordinat arası uzaklığı döndürür.
  closestMultiple: (current,aim,dir) => {
    current = allFuncs.toInt(current);
    while (current%aim!=0){
      current+=dir
    }
    return current;
  },// Bu fonksiyon girilen değere en yakın kendisinden küçük veya büyük girilen katsayının katını döndürür.
  toInt: (n) => n-n%1, //Bu fonksiyon girilen sayının tam kısmını  döndürür
  runForArr: (arr,funcName) => {
      for(let i = 0; i < arr.length; i++){
          eval("arr[i]."+funcName+"();")
      }
  },//Bu fonksiyon girilen bir dizi için girilen fonksiyonu gerçekleştirir
  percit: (n, start1, stop1) => ((n - start1) * 100) / (stop1 - start1),
  //Bu fonksiyon girilen değerin yine girilen başlangıç ve son değerleri arasında yüzde kaçta olduğunu döndürür. (23. satır)
  choice: (arr) => {
    let len = Math.floor(Math.random()*(arr.length))
    return arr[len]
},//Bu fonksiyon verilen diziden rastgele bir eleman döndürür
  probability: (percentage,will,willnot) => {
  let f = []
  let t = []
for (i = 0; i < 100-percentage;i++){
  f.push(willnot)
}
for (i = 0; i < percentage;i++){
  t.push(will)
}
  return allFuncs.choice(f.concat(t))
},//Bu fonksiyon verilen yüzde ihtimali ile 2.parametreyi döndürür. Eğer 2. parametre seçilmezse 3. parametre döndürülür.

  ortalama: (...args) => {
  var sum = 0;
    for (let i = let i: number ngth;i++){
    sum += args[i]
    }
  return sum / args.length
},//Bu fonksiyon girilen sayıların ortalamasını döndürür.

haveCommon: (arr1,arr2) => arr1.some(elem => arr2.includes(elem)), //Bu fonksiyon iki dizi arasında ortak eleman varsa true yoksa false döndürür.

Vektor: class {
  constructor(x, y) {
      this.x = x;
      this.y = y;
      this.type = "vector"
  };
  ekle(v) {
      if (!v.type || v.type != "vector") {
          throw "TypeError: add function requires a vector"
      }else {
          this.x = this.x+v.x;
          this.y = this.y+v.y;
          return this;
      }
      };
},//Bu nesne kod içinde kullanılan pozisyon, yön gibi vektörler için yazıldı.
randomFloat: (min,max) => Math.random() * (max-min) + min //Bu fonksiyon girilen iki sayı arasından rastgele bir ondalık veya tamsayı döndürür.
}
```

## *Fonksiyonlar 2:*

## Sonuç ve Tartışma

Her ne kadar simülasyon modelleme yönteminin altında yatan varsayımlar yetersiz olsa da ve bazen incelenmekte olan sistem için uygun olmasa da, araştırma konusu olan sistem hakkında bazı bilgilere ve bunlar arasındaki olası etkileşimlere, ayrıca önceki biyokimyasal ve genetik bilgilere dayanarak, kaba bir modelle başlamamızı ve bu kaba model ile canlıların (örneğin hücre) yapısı, davranışları hakkında bize en azından sıfırıncı dereceden bir hipotez sağlar. (16).

Elbette doğaya, gerçek hayata baktığımız zaman neredeyse sayısız diyebileceğimiz etmenlerin, değişkenlerin, çok ince ayarların olduğunu biliyoruz. Sıcaklık, nem seviyesi, hastalıklar, afetler, bilinç gibi gerçekten sayısız süreç ve etmen var ve tabii ki bunu bir bilgisayar programına aktararak simüle edebilmek, düzenlemek neredeyse imkânsızdır. Çok önemli üniversiteler ve enstitüler bu konuda çalışmalar düzenlense bile tamamen gerçekçi diyebileceğimiz simülasyonların şu anda dünyada bulunmadığını biliyoruz. En azından kompleks canlılar için bunu söyleyebiliriz. Bu yüzden tamamen gerçekçi, mükemmel diyemeyiz ama bu projenin çok önemli bir adım olduğu kanaati ifade edilebilir. Bunu, programı çalıştırdığımızda yaptığımız gözlemler ve kazanımlarla da kanıtlayabiliyoruz.

Projenin önemli kısmı tamamlandıktan sonra, öğrenciler, çeşitli eğitim düzeylerinde bireyler ve bilişim, biyoloji konularında uzman kişilerle paylaşıldı, görüş ve eleştirileri alındı. Açıkçası paylaşılan herkes projeyi çok etkileyici ve anlamlı buldu. En çok beğenilen kısım ise çok kolay anlaşılır olmasıydı.

## Öneriler

Projeye ilk olarak, kullanıcı arayüzü eklenebilir.

Projenin daha sonraki aşamasında, bu canlıların davranışlarını ve özelliklerini daha kompleks hale getirmek mümkün olabilir ve ayrıca daha fazla canlı ve ortam özelliği eklenebilir. Örnek vermek gerekirse sulu bölgeler olabilir veya simülasyona farklı iklim modelleri eklenebilir ve proje ilerledikçe modellemenin temsil düzeyi artarak gerçek hayata daha da yakınlaşabilir.

Bu çalışmaya daha fazla etmen, değişken eklemek ve daha fazla optimize etmek mümkündür. Optimizasyon çok önemli, çünkü kod arttıkça ve bilgi, veri eklendikçe, simülasyon o kadar kod hatalarına açık ve yavaş hale gelebilir. Bu yüzden bu çalışmayı geliştirmek için hem çok daha fazla veri eklemek hem de aynı zamanda kodu çok daha hızlı ve optimize hale getirmek gerekmektedir.

**Kaynaklar**


1) Brodland, G. W. (2015). How computational models can help unlock biological systems. Seminars in Cell & Developmental Biology. Çevrimiçi ön yayın. https://doi.org/10.1016/j.semcdb.2015.07.001

2) Hogeweg, P. (2011). The Roots of Bioinformatics in Theoretical Biology. PLoS Comput Biol 7(3): e1002021. Çevrimiçi ön yayın. https://doi.org/10.1371/journal.pcbi.1002021

3) Bataw, S. (2016). Siyez (Triticum monococcum ssp. monococcum) buğdayında korunmuş mikro malarin yüksek verimli dizileme ve simülasyon analizleriyle tanımlanarak karakterize edilmeleri. (Yayımlanmamış yüksek lisans tezi). Abant İzzet Baysal Üniversitesi/Fen Bilimleri Enstitüsü Erişim adresi: https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

4) Kül, Ö. (2009). Trombinin yavaş ve hızlı formlarının konformasyon değişimlerinin moleküler dinamik simülasyonu ile incelenmesi. (Yayımlanmamış yüksek lisans tezi). Hacettepe Üniversitesi/ Fen Bilimleri Enstitüsü, Ankara. Erişim adresi: https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp

5) Dev, O. B., Aydogan, R. ve Öztop, E. (2017). Yapay yaşam simülasyonunda hayatta kalmak. 25. Signal Processing and Communications Applications Konferansında sunulan bildiri, Antalya. Çevrimiçi ön yayın. https://doi.org/10.1109/SIU.2017.7960611

6) Karr, J. R., Sanghvi, J. C., Macklin, D. N., Assad-Garcia, N., Glass, J. I. ve Covert M. W. (2012). A Whole-Cell Computational Model Predicts Phenotype from Genotype. Çevrimiçi ön yayın. https://doi.org/10.1016/j.cell.2012.05.044

7) Traore, M. (2018). Modélisation cellulaire et simulation physique : contribution à l'analyse de la dynamique de population des insectes ravageurs. Autre [cs.OH]. (Yayımlanmamış doktora tezi). Université de Bretagne occidentale, Brest. Erişim adresi: https://tel.archives-ouvertes.fr/tel-01880972/

8) Mount, G. A., Haile, D. G., Davey, R. B. ve Cooksey, L. M. (1991). Computer Simulation of *Boophilus* Cattle Tick (Acari: Ixodidae) Population Dynamics. Journal of Medical Entomology, Volume 28, Issue 2, Sayfa 223–240. Çevrimiçi ön yayın. https://doi.org/10.1093/jmedent/28.2.223

9) Virtual Liver (2014, Ekim 13). Retrieved from https://www.h-its.org/projects/virtual-liver/

10) Downey, A. B. (2017). *Modeling and Simulation in Python*. Needham, Massachusetts, Green Tea Press. Retrieved from https://greenteapress.com/wp/modsimpy/

11) Jacobson, I. (1992). *Object-Oriented Software Engineering*. Retrieved from https://archive.org/details/objectorientedso00jaco/page/43

12) Nieuwenhuys, E., Otto, J. (2017). Peacock spider, *Maratus volans.* Erişim adresi: https://ednieuw.home.xs4all.nl/australian/salticidae/Peacock_spider_Maratus_volans.htm

13) Otto, J.C. ve Hill, D.E. (2011). An illustrated review of the known peacock spiders of the genus Maratus from Australia, with description of a new species (Araneae: Salticidae: Euophryinae) [PDF File]. Retrieved from https://peckhamia.com/peckhamia/PECKHAMIA_96.1.pdf

14) Girard, M. B., Kasumovic, M. M., & Elias, D. O. (2011). Multi-modal courtship in the peacock spider, Maratus volans (O.P.-Cambridge, 1874). *PloS one*, *6*(9), e25390. Çevrimiçi ön yayın. https://doi.org/10.1371/journal.pone.0025390

15) Greshko, M. (2015, December 1). Female Peacock Spiders Underwhelmed By Disco-Dancing Suitors. National Geographic Erişim adresi: https://www.nationalgeographic.com/news/2015/12/151201-australia-peacock-spider-colorful-courtship-sex-animals-science/

16) National Research Council. (2005). *Catalyzing Inquiry at the Interface of Computing and Biology. Computational Modeling and Simulation as Enablers for Biological Discovery*. (117-204). Washington, DC: The National Academies Press. Retrieved from https://www.nap.edu/read/11480/chapter/7

**Ekler**

Ek - 1 : Proje Ekran Görüntüleri

Not: Ekler EK BELGELERE yüklenmiştir.